Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

SERVIDOR DE MOBILIDADE

José Miguel Coelho dos Santos Rangel

Mestrado em Engenharia Informática

2007

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

SERVIDOR DE MOBILIDADE

José Miguel Coelho dos Santos Rangel

Projecto orientado pelo Prof. Dr.º Carlos Lourenço
e co-orientado por Eng. Nuno Rodrigues

Mestrado em Engenharia Informática

2007

Declaração

José Miguel Coelho dos Santos Rangel, aluno nº 28345 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Servidor de Mobilidade", realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 23 de Outubro de 2007

Nuno Miguel Pombo Rodrigues, supervisor do projecto de *José Miguel Coelho dos Santos Rangel*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "Servidor de Mobilidade".

Lisboa, 23 de Outubro de 2007

Resumo

Este documento descreve o projecto realizado no ano lectivo 2006/07 no âmbito da disciplina Projecto em Engenharia Informática do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa. Neste projecto fui orientado pelo Prof. Dr.º Carlos Lourenço em representação da faculdade e pelo Eng.º Nuno Rodrigues em representação da instituição de acolhimento.

O projecto em que fui inserido e sobre o qual este documento se irá debruçar tem como título "Servidor de Mobilidade", e foi desenvolvido nas instalações da GEDI (Gabinete de Estudos e Divulgação Informática) em Miraflores.

A principal actividade da GEDI é a concepção, desenvolvimento e implementação de soluções informáticas, nomeadamente em diferentes áreas da Administração Pública. Recentemente a GEDI iniciou o desenvolvimento de uma nova geração de software denominada SIAG (Sistema Integrado de Apoio à Gestão). Esta é uma solução global, verdadeiramente integrada, que abrange diversas áreas tais como, a gestão financeira, patrimonial, pessoal, fluxos de informação, etc. A cada área corresponde um módulo na aplicação, sendo estes parametrizáveis consoante os requisitos do cliente.

Uma das características do SIAG é a existência do módulo *Mobile*, responsável por oferecer uma interface e um conjunto de funcionalidades adaptadas às características dos dispositivos móveis. Coube-me a responsabilidade de implementar a totalidade deste módulo, sendo supervisionado pelo Gestor de Projecto o Eng.º Nuno Rodrigues.

O browser utilizado neste projecto foi o *PIE (Pocket Internet Explorer)*. Numa primeira fase o projecto focou-se na análise de todas as funcionalidades que o browser oferece, e quais as suas limitações comparativamente com os browsers disponíveis num computador pessoal. Esta fase foi importante para perceber quais das tecnologias utilizadas no SIAG "tradicional"não podem ser transpostas para o módulo *Mobile* e quais as alternativas.

Concluída esta primeira fase, deu-se inicio ao desenvolvimento de protótipos que implementam algumas operações do SIAG. Estes protótipos são extremamente importantes pois permitem analisar várias soluções, ajudando assim a decidir qual a melhor forma de implementar o produto final. Recolhida toda esta informação foi então possível avançar para o desenvolvimento da solução final. De forma a assegurar a validade desta solução, todo o trabalho de desenvolvimento foi complementado por testes.

Abstract

The present thesis contains the research project developed during the school year 2006/2007 to achieve the master's degree on Computer's Science held by the *Faculdade de Ciências da Universidade de Lisboa*. The research project was coordinated by the Professor Carlos Lourenço from the *Faculdade de Ciências* and by the Engineer Nuno Rodrigues representing GEDI (*Gabinete de Estudos e Divulgação Informática*), the host institution.

This research project was developed at GEDI's office in Miraflores and it is focused on a Mobility Server which is also the thesis title.

GEDI's main activity is the development of software solutions, specialy for the public sector. It has recently started the developing of a new kind of software named SIAG which stands for *Sistema Integrado de Apoio à Gestão*. This software is a global solution, covering different areas like financial management, human resources, etc. Each one of the distinct areas has an independent module in the application, every module can be customized to fulfil the customer requirements.

One of SIAG characteristics is the Mobile module, responsible for adapt and deliver information to a mobile device. The objective of this project was to implement the Mobile module. The team responsible for the SIAG solution has 10 elements. My work was supervised by the Engineer Nuno Rodrigues.

The internet browser used in this project was the Pocket Internet Explorer. The first step of the project was the study of the browser capabilities and make a comparison with the common browsers that are executed in PC's. This overview was important to understand wich tecnologies used in SIAG could be recycled into the Mobile, and what were the alternatives.

Finished this first objective, began the development of simple prototypes that implemented some operations from SIAG. This prototypes are extremely useful, because they allow the analysis of diferent solutions. Helping to decide the better way to deliver the final product. With all this information the project moved to the development of the final soluction. Every feature was tested to assure the correctness of the application.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo serve de breve apresentação ao projecto, ao trabalho desenvolvido e à estrutura deste documento.

1.1 Contextualização do Projecto

A informação é um recurso importantíssimo na era actual, numa economia onde as empresas competem ferozmente pela sua sobrevivência, a capacidade de retirar mais valias da informação ao seu dispor é fundamental.

As organizações, de forma a conseguir obter o máximo proveito de toda a informação reunida ao longo da sua actividade, fizeram grandes investimentos em aplicações de logística, *CRM (Customer relationship management)* e *ERP (Enterprise Resource Planning)*. Toda esta capacidade de tratamento de informação, está apenas disponível aos colaboradores da empresa no seu local de trabalho. Cada vez mais as organizações têm os seus colaboradores a trabalhar no terreno em contacto directo com o cliente. Nestas situações não conseguem tirar partido do grande investimento realizado pela empresa em aplicações de negócio. A enorme volatilidade da informação obriga a que as actualizações sejam disponibilizadas rapidamente a quem necessita de estar o mais bem informado possível, de forma a conseguir maximizar a sua produtividade. Actualmente este é um dos maiores desafios das organizações.

Com o desenvolvimento da tecnologia é hoje possível aceder à informação presente nos sistemas da nossa empresa com um simples telemóvel ou *PDA (Personal Digital Assistant)*, preenchendo desta forma a lacuna que existe em termos de trabalho no campo. Os colaboradores passam a ter disponível uma forma de interagirem e reagirem à volatilidade da informação.

Os equipamentos disponíveis, apesar de bastante avançados e com uma capacidade de processamento apreciável, possuem limitações em relação aos poderosos computadores pessoais. Começando pelo tamanho do ecrã, não se pode pensar em disponibilizar num dispositivo móvel os conteúdos apresentados da mesma forma que

num computador pessoal. Outra característica a ter em atenção são os dispositivos de input bastante limitados nos dispositivos móveis.

É necessário mobilizar as aplicações de negócio já existentes nas empresas, para que estas possam ser acedidas através de um dispositivo móvel por quem trabalha no exterior e necessita de ter acesso à informação. Desta forma consegue-se tirar o máximo partido do investimento realizado pelas organizações em aplicações de negócio.

1.2 Descrição do Projecto

1.2.1 SIAG

A solução SIAG, desenvolvida pela GEDI consiste numa aplicação de *ERP* pensada especialmente para a Administração Pública. As instituições enfrentam actualmente um grande desafio com a necessidade de se preparem para uma administração pública moderna e tecnologicamente evoluída. A GEDI faz uso do seu vasto conhecimento da realidade da Administração Pública Portuguesa adquirida ao longo de mais de uma década de trabalho conjunto e assume o desenvolvimento de uma nova geração de software.

Este produto tem como objectivo ser uma plataforma de apoio à gestão financeira, patrimonial, pessoal, fluxos de informação, etc. Os módulos presentes na aplicação SIAG são Planeamento do Processo de Gestão, Produtos e Existências, Compras, Vendas e Receitas, Gestão do Imobilizado, Recursos Humanos, Gestão Financeira-Controlling, Tesouraria, Informação-Workflow-Arquivo, Reporting, Relação com o Meio Envolvente e Utilitários e Sistema. Todos estes módulos formam uma solução integrada, que incorporando diversos mecanismos de configuração e parametrização permite dar resposta às várias necessidades e objectivos dos diferentes tipos de organismos.

O SIAG é uma aplicação web, bastando assim um simples *browser* para aceder à aplicação num ambiente web em intranet ou internet. A aplicação é desenvolvida fazendo uso de um vasto leque de algumas das mais poderosas tecnologias actuais, tais como *Java*, *JSP (Java Server Pages)*, *Hibernate*, *Struts* e *Spring*. Todas estas tecnologias são baseadas em formato aberto. Esta opção de implementação permite ao SIAG interagir com as plataformas standard existentes no mercado tanto ao nível de SGBDs (Sistema de Gestão de Base de Dados) como de servidores de aplicações. O que se traduz numa grande vantagem para o produto final, uma vez que as soluções já adquiridas pelo cliente ao nível de SGBD e servidores de aplicações podem ser utilizadas com o SIAG.

1.2.2 Módulo Mobile

O projecto onde fui inserido tem como objectivo desenvolver um servidor de mobilidade para o SIAG. Tratando-se de uma aplicação web, pode ser acedido a partir de qualquer dispositivo que disponha de um *browser* e de ligação à Internet. A necessidade do desenvolvimento de um servidor de mobilidade advém do facto de a interface disponibilizada pelo SIAG ser bastante complexa, devido às vastas funcionalidades que disponibiliza aos utilizadores. Uma interface deste género não se encontra optimizada para o ecrã de um dispositivo móvel. Outro problema que se levanta é o facto de a interface original fazer um grande uso de *Java Applets* e *JavaScript*, tecnologias que não são compatíveis com a especificação do *PIE*, *browser* que integra o *Windows Mobile*, sendo este o sistema operativo dominante no mercado em termos de *PocketPC* e telemóveis de 3ª geração.

Dado as limitações do *PIE* é pois necessário criar um sistema que permita gerar a apresentação dos conteúdos optimizada para dispositivos móveis, é esta a função do servidor de mobilidade. Este componente será parte integrante do SIAG, tendo como função, detectar se o cliente que está a comunicar com o servidor se trata de um dispositivo móvel ou um *browser* comum. Consoante o tipo de cliente os conteúdos serão gerados de acordo com as suas capacidades. O principal desafio deste módulo *Mobile* é desenvolver uma camada de apresentação optimizada para *browsers* de dispositivos móveis, utilizando as camadas de negócio e de dados já desenvolvidas.

Este projecto vai além da uniformização de uma interface que permita apresentar conteúdos optimizados para dispositivos móveis. Outro grande objectivo passa por conseguir que esta interface seja completamente dinâmica. Ou seja, o utilizador do SIAG deve poder a qualquer momento configurar de acordo com as suas necessidades a interface ao seu dispor. A estrutura dos menus e sua navegação, deve também acompanhar toda a natureza dinâmica desta solução. Para que se possa rentabilizar ao máximo os recursos disponíveis no dispositivo móvel, deve-se exibir ao utilizador apenas o que ele desejar e na forma que lhe convier.

1.3 Trabalho desenvolvido

Durante as primeiras duas semanas ocorreu o período de adaptação à empresa e ao projecto que me foi atribuído. Numa primeira fase foi feita uma pesquisa sobre o tema do projecto e analisados vários produtos existentes no mercado. Esta fase culminou com a elaboração de um documento com as conclusões e a sua apresentação. A apresentação foi realizada perante o supervisor Eng.º Nuno Rodrigues e o administrador Eng.º Carlos Alves.

Esse trabalho de pesquisa ajudou a compreender quais os objectivos do projecto

e perceber que tecnologias são utilizadas no desenvolvimento de aplicações deste género.

Terminado o período de adaptação e já com uma ideia sobre o tema, foi implementada uma simples página optimizada para dispositivos móveis, utilizando apenas *HTML (Hypertext Markup Language)*. Após este primeiro passo, comecei a explorar a tecnologia *JSP* e a criar páginas dinâmicas.

Chegada a quarta semana iniciei o desenvolvimento do primeiro protótipo enquadrado com o SIAG. O primeiro objectivo seria descobrir se os pedidos que chegam ao servidor eram originados por um dispositivo móvel ou por um *browser* tradicional. Após este passo o pedido é direccionado para a página de autenticação original ou para a de autenticação móvel. Depois de implementada a capacidade de detecção do tipo de pedido que chega ao servidor, implementou-se uma página de autenticação para dispositivos móveis.

A usabilidade numa interface para dispositivos móveis é uma questão muito importante, este aspecto levou à realização de um levantamento de interfaces existentes, focando os objectivos em ecrãs de input e de output de informação.

Nesta fase do projecto e já com alguns conhecimentos sobre o enquadramento do mesmo e tecnologias necessárias ao seu desenvolvimento, iniciou-se a implementação de alguns dos componentes da versão final.

O primeiro componente a ser implementado designa-se *Cockpit*. Este é o esqueleto externo da interface, todos os ecrãs gerados são incluídos neste componente. No *Cockpit* é gerada a estrutura de menus das aplicação e barra de navegação. A zona para as mensagens de erro também está contemplada neste componente. Depois de concluída esta implementação, iniciou-se a do *Layout SIAG móvel*. Esta é a designação da *JSP* que é incluída no *Cockpit* e é a base de todos os ecrãs que contêm formulários e dos que exibem listas de registos. É aqui que é apresentado o título e são desenhados os botões consoante o tipo de operação.

Depois de toda a estrutura da interface implementada, inicia-se o desenvolvimento das funcionalidades que vão permitir mobilizar o SIAG. Começando pelo ecrã de outuput do tipo lista, o qual lista um conjunto de registos. O ecrã seguinte foi o de input, este permite visualizar um registo em detalhe, alterar ou inserir um novo. Ambos os ecrãs são completamente dinâmicos, o que permite que qualquer módulo do SIAG os possa utilizar.

Com este conjunto de funcionalidades básicas já implementadas, a próxima fase consistiu em melhorar estas capacidades introduzindo uma maior complexidade ao nível da interface e das operações. Os ecrãs de input foram refinados com um maior nível de detalhe, aproximando a interface móvel da interface de *browser*.. Procedeu-se à implementação de mecanismos de pesquisa e ordenação, imprescindíveis para optimizar a área disponível para apresentação de resultados.

A última fase de desenvolvimento do projecto, envolveu a criação de um mecanismo de desenho de ecrãs para o dispositivo móvel. Os ecrãs de input da interface são totalmente configuráveis, cabe ao utilizador decidir qual a apresentação do ecrã. Isto é, decidir quais os campos que devem ser exibidos e qual o seu tipo. Esta funcionalidade de desenho foi implementada em ambas as interfaces (móvel e tradicional).

1.4 Enquadramento institucional

A GEDI é uma empresa com mais de duas década de existência que tem actualmente como principal actividade o desenvolvimento de software de apoio à actividade da Administração Pública. Esta actividade representa mais de 90% do seu negócio. Com mais de uma década de experiência no desenvolvimento de soluções para organismos tão variados como Universidades, Institutos Politécnicos, Direcções Gerais, Secretarias Gerais, Secretarias Regionais, Câmaras Municipais, Escolas Secundárias, etc. A GEDI conta já com as seguintes aplicações informáticas:

- ☞ Gestor - Gestão Orçamental Contabilidade Pública
- ☞ Prep'OE - Elaboração de Projectos de Orçamento
- ☞ RH+ - Gestão de Pessoal
- ☞ SPID - Sistema de Processamento de Deslocações
- ☞ Isys - Gestão de Informação Expediente e Arquivo
- ☞ Facturação - Facturação de Bens e Serviços
- ☞ Stocks - Gestão de Stocks

O SIAG faz parte de uma nova geração de software da GEDI. Com o objectivo de ser uma solução verdadeiramente integrada para responder à necessidade de inovação cada vez mais acelerada a nível dos processos, dos produtos e dos serviços, com os consequentes impactos a nível da gestão e das estruturas organizacionais.

Outras das actividades da GEDI são o serviço de Consultadoria, Formação e Assistência Técnica. Todos estes serviços são orientados para as soluções desenvolvidas pela GEDI e funcionam como complemento às mesmas.

1.5 Organização do documento

Este documento está organizado da seguinte forma:

- ☞ Capitulo 1 - Introdução

- ☞ Capitulo 2 - Framework GEDI
- ☞ Capítulo 3 - Servidor de Mobilidade
- ☞ Capítulo 4 - Metodologia e Planeamento
- ☞ Capítulo 5 - Trabalho Realizado
- ☞ Capítulo 6 - Conclusão

Capítulo 2

Framework GEDI

A GEDI fruto do seu historial no desenvolvimento de aplicações informáticas para a área de gestão, acumulou um vasto conhecimento que hoje é aproveitado para optimizar o seu processo de desenvolvimento de software.

Esta nova geração de software desenvolvida pela GEDI, da qual se destaca o SIAG, assenta numa arquitectura bem definida. Com este modelo qualquer aplicação é implementada utilizando uma *framework* conhecida e de valor comprovado (*Framework* GEDI).

2.1 Arquitectura

A *Framework* GEDI segue o modelo de *Arquitectura em três camadas*[15]. Este modelo demarca os três principais componentes de uma aplicação cliente-servidor. Cada uma destas camadas pode desta forma ser desenvolvida e mantida separadamente das restantes, permitindo o desenvolvimento com recurso a tecnologias distintas. No caso da *Framework* GEDI o modelo seguido e tecnologias utilizadas é apresentado na figura 2.1:

- **Camada de Apresentação** (*UI Layer*) Esta camada que se situa no nível superior da arquitectura é responsável por todo e qualquer conteúdo que esteja visível ao utilizador. As tarefas delegadas a esta camada incluem gerir os pedidos e respostas resultantes da interacção com o sistema, providenciar a delegação de chamadas à lógica de negócio, disponibilizar os dados de uma forma coerente ao utilizador e executar validações de introdução de dados.

- **Camada de Negócio** (*Business Layer*) Situada no nível intermédio, esta camada é responsável pela lógica e validação das regras de negócio da aplicação, gestão transaccional e das dependências entre objectos de negócio. Processa toda informação que flui entre as duas camadas envolventes através das interfaces definidas para esse efeito.

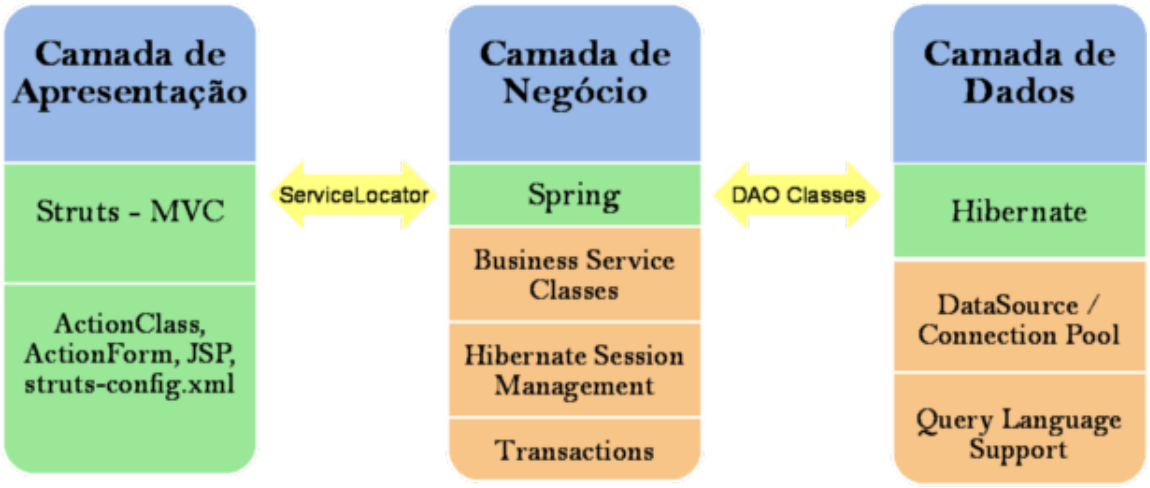

Figura 2.1: Modelo da Arquitectura GEDI

☞ **Camada de Dados** (*Persistence Layer*) A responsabilidade da gestão de todos os dados da aplicação está nesta camada. A informação relacional contida na base de dados é aqui transformada em objectos, que por sua vez são passados à camada superior, abstraindo assim das especificidades das base de dados presentes no sistema. As operações de criação, actualização e remoção da informação presente na base de dados são efectuadas por esta camada.

2.2 Tecnologias utilizadas

A implementação da solução SIAG utiliza algumas das mais importantes tecnologias actuais, capazes de acrescentar valor ao desenvolvimento e ao produto final.

2.2.1 Java

A aplicação SIAG é desenvolvida utilizando a linguagem *Java*. O Java é uma linguagem de programação orientada a objectos, tem como principal vantagem ser independente da plataforma de execução. Esta independência permite que a aplicação seja executada nos mais populares sistemas operativos.

2.2.2 HTML

HTML é uma linguagem de marcação utilizada para produzir páginas web.

2.2.3 JavaScript

JavaScript é uma linguagem de programação bastante utilizada no desenvolvimento de páginas web. O código *JavaScript* é executado na parte cliente da aplicação e é bastante utilizado para efectuar validações de formulários. Pode também ser utilizado para interagir com objectos de outras aplicações.

2.2.4 Apache Struts

O *Struts*[14] é uma *framework* de código livre para aplicações web desenvolvidas em *Java* que verifica o bem conhecido padrão de desenho *MVC (Modelo-Vista-Controlo)*[7, 8] que divide o processamento em três componentes distintas:

- ☞ Modelo

  O Modelo representa os dados da aplicação que neste caso são os *pojos (Plain Old Java Object)* que são obtidos através dos serviços da camada de negócio.

- ☞ Controlo

  O Controlo é conseguido através do mapeamento de *Actions*. Uma *Action* é o ponto de entrada na aplicação e é onde se trata a lógica para determinada acção. As *Actions* pertencem à camada de apresentação da aplicação e interagem com a camada de negócio através de serviços. Cada *Action* tem designado um serviço que é responsável por tratar da sua lógica de negócio.

- ☞ Apresentação

  A Apresentação é feita pelo intermédio de *JSPs*. Os dados que são apresentados na *JSP* são obtidos através de *ActionForms* que são construídos na componente de controlo e que são mapeados para a *JSP* de forma automática pelo *Struts*. Os *ActionForms* são objectos *Java* que mapeiam uma propriedade para cada campo do formulário *HTML*.

A configuração da componente de *Struts* é feita num ficheiro *XML (Extensible Markup Language)* designado *struts-config.xml*. Este indica para cada uma das acções que pode ser submetida ao servidor, qual a classe responsável por tratar o pedido, bem como um conjunto de redireccionamentos possíveis consoante a acção foi bem sucedida ou existam diferentes apresentações. Tem de se indicar também qual o *ActionForm* que representa o formulário que vai ser apresentado ao utilizador. O modelo de interacção do *Struts* está representado na figura 2.2.

2.2.5 Tiles

Tiles[1] é um componente da *framework Struts*. Permite dividir a página a apresentar num conjunto de zonas que podem ser definidas através de um ficheiro *XML*. No

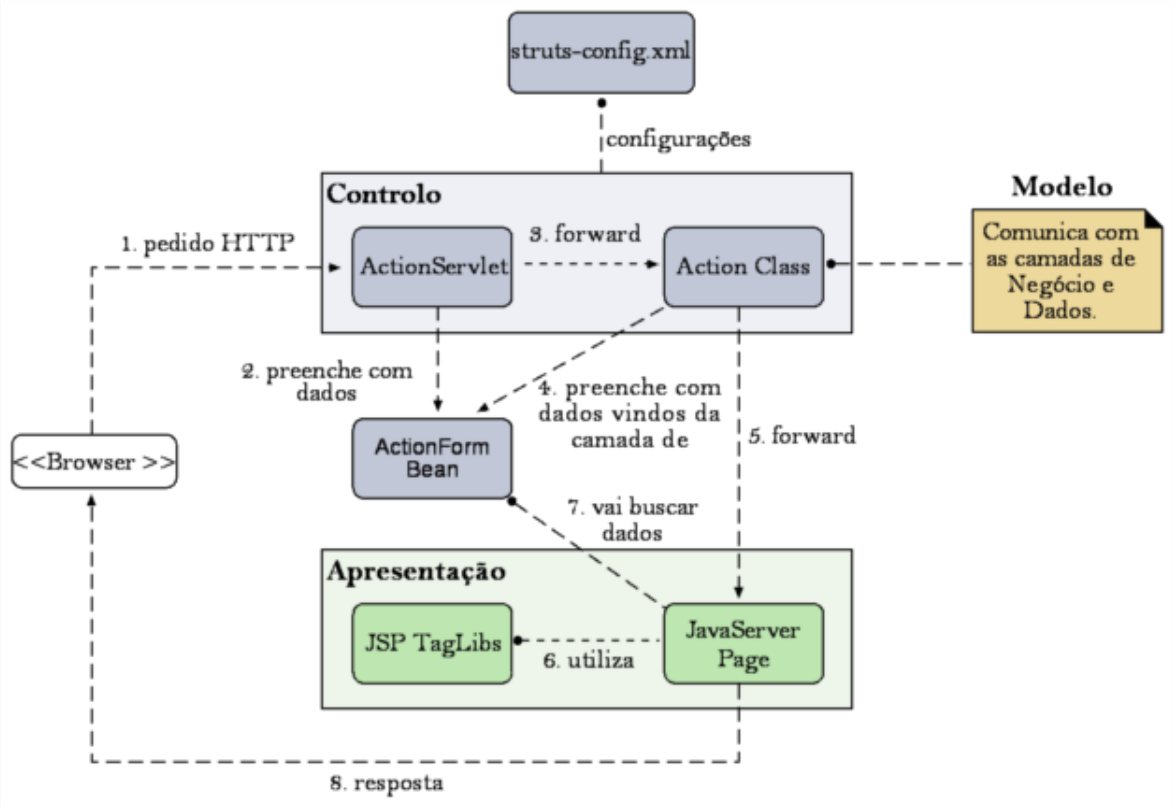

Figura 2.2: Modelo Struts

caso do SIAG é utilizado para definir uma página de formulário com funcionalidade de base que é partilhada por todos os outros e onde é adicionado o titulo da página, o endereço para onde o formulário deve ser submetido e qual a *JSP* a incluir. Esta *JSP* contem a configuração específica de cada formulário de ecrã.

2.2.6 Java Server Pages (JSP)

Java Server Pages[6] é uma tecnologia Java que permite gerar conteúdo *HTML* dinamicamente em tempo de execução. Através do uso de etiquetas *JSP* onde pode ser inserido código Java e de um servidor de aplicações que suporte *JSP* e *Servlets* consegue-se criar páginas *HTML* dinâmicas. Quando o servidor de aplicações recebe o pedido de uma página *JSP* o conteúdo das etiquetas presentes no ficheiro é compilado e o seu resultado é código *HTML*.

2.2.7 Spring

Spring[16] é uma *framework* que trata de instanciar objectos Java e respectivas relações através da sua configuração em *XML*. Isto permite limpar o código e recorrer apenas a ficheiros *XML* para definir as relações entre objectos tornando o código mais limpo. Permite também definir mecanismos que interceptam as cha-

madas aos objectos para permitir um pré-processamento e depois redireccionam a chamada para o objecto real. No SIAG o *Spring* é utilizado para instanciar serviços, bastando para isso aceder ao objecto de contexto de *Spring* e pedir determinado serviço passando o nome lógico. O nome lógico é aquele que é dado na configuração do serviço e pelo qual é feita a sua procura.

2.2.8 Hibernate

O *Hibernate*[10, 5] mapeia as tabelas de uma base de dados, no esquema relacional para o modelo orientado a objectos, através da introdução de objectos *Java* designados por *Pojo (Plain Old Java Objects)* que representam registos. A relação que existe entre tabelas também é transposta para os objectos através de relações muitos-para-um onde o *Hibernate* acrescenta ainda a possibilidade de ter relações um-para-muitos através de conjuntos de *pojos*. Ex: Um grupo tem uma relação um-para-muitos com vários utilizadores. Estes *pojos* são carregados do SGBD e são mantidos ligados à sessão *Hibernate*. A sessão consegue detectar se existem alterações nos pojos de modo a que seja necessário efectuar uma actualização ao registo no SGBD. Permite também a criação de novos registos bastando indicar à sessão que guarde determinado *pojo* no SGBD. Existe a noção de *cascade* para as relações um-para-muitos nas gravações de *pojos*, onde apenas precisamos de indicar a gravação do *pojo* de topo e os conjuntos de *pojos* relacionados vão ser persistidos através do mecanismo de *cascade*.

Esta *framework*, além de mapear as tabelas de uma base de dados em objectos Java, também disponibiliza funcionalidades que permitem efectuar interrogações e tratar os resultados de uma forma transparente, independentemente do SGBD utilizado. O *Hibernate* gera as chamadas *SQL (Structured Query Language)* e trata do processamento dos resultados. Este modo de funcionamento faz com que a aplicação possa ser executada sobre todos os SGBD *SQL*. Na figura 2.3 está representada a arquitectura de alto nível da *framework Hibernate*.

2.3 Estrutura dos Projectos

Os projectos que sejam desenvolvidos tendo por base a *Framework* GEDI obedecem a uma estrutura bem definida. Todos eles são desenvolvidos em cima de um projecto denominado **Base**.

2.3.1 A Base

O projecto Base implementa todas as funcionalidades das camadas de Apresentação, de Dados e uma parte da camada de Negócio responsável pela estrutura de Utili-

Figura 2.3: Framework Hibernate

zadores e Privilégios das soluções. Para qualquer solução informática que a GEDI pretenda implementar é criado um novo projecto, este é desenvolvido sobre a Base. Toda a lógica de negócio inerente à nova solução é implementada neste projecto.

A solução SIAG enquadra-se nesta descrição, é um projecto montado sobre o projecto Base que implementa apenas a camada de Negócio. Todas as funcionalidades das restantes duas camadas estão implementadas na Base. Este modelo permite que o desenvolvimento de novas soluções apenas se foque na correcta implementação e validação dos requisitos do negócio.

2.3.2 O Mobile

O servidor de mobilidade também implicou a criação de um novo projecto, ao qual foi atribuido o nome ***Mobile***. Este seguiu a mesma estrutura de todos os outros, ou seja, também foi implementado sobre a Base. Contudo, o *Mobile* não representa uma nova solução, trata-se de um novo módulo a acrescentar à *framework*. O trabalho desenvolvido neste projecto é orientado para a camada de Apresentação e não para a de Negócio.

A utilização comum do projecto Base e a clara separação entre os restantes projectos torna-os completamente independentes entre si. Uma instalação de SIAG não necessita do *Mobile* para funcionar, mas caso se pretenda mobilizar a aplicação SIAG basta juntar à instalação o projecto *Mobile* e todas as suas funcionalidades

ficarão disponíveis sem qualquer necessidade de configuração. O exemplo dado de integração com o SIAG é válido para qualquer outra solução que seja desenvolvida utilizando a *Framework* GEDI.

O *Mobile* essencialmente acrescenta novas funcionalidades à camada de apresentação implementada pela Base. Como as soluções desenvolvidas implementam apenas a camada de Negócio, a inclusão de funcionalidades para dispositivos móveis é opcional e completamente independente da solução final.

Capítulo 3

Servidor de Mobilidade

Este capítulo descreve os objectivos do Servidor de Mobilidade e as limitações tecnológicas encontradas no seu desenvolvimento.

3.1 Objectivos

A ideia de desenvolver um servidor de mobilidade surge da necessidade de proporcionar aos elementos de cada organização uma maior disponibilidade no acesso à aplicação, independentemente do local onde se encontram e dos recursos disponiveis. O mundo evolui cada vez mais para um estado de constante movimento de pessoas e serviços, é o objectivo deste género de soluções acompanhar esta evolução.

O SIAG segue esta filosofia, para atingir o fim desejado é necessário implementar uma componente que se responsabilize por disponibilizar os conteúdos a dispositivos móveis. Os principais objectivos do módulo *Mobile* do SIAG são:

- ☞ Integrar a solução na estrutura de privilégios existentes
- ☞ Disponibilizar todos os componentes de negócio existentes na aplicação original
- ☞ Permitir consultar listas de registos de todos os componentes de negócio
- ☞ Criar um mecanismo que permita criar e/ou editar registos de qualquer componente
- ☞ Permitir que o utilizador adapte a solução às suas necessidades
- ☞ Utilizar ao máximo os mecanismos já existentes na aplicação base
- ☞ Tornar a existência do módulo *Mobile* totalmente transparente e independente da aplicação final

3.2 Limitações Tecnológicas

Os dispositivos móveis que existem actualmente, apesar do seu reduzido tamanho, apresentam um poder de processamento e de memória considerável. Estas capacidades possibilitaram que algumas aplicações que nos habituámos a utilizar nos computadores pessoais tenham sido adaptadas para os dispositivos móveis. Tendo como referência os modelos *PocketPC*, equipados com o sistema operativo *Windows Mobile*, são várias as aplicações que se enquadram nestas condições tais como *Word*, *Excel*, *Outlook*, *Internet Explorer* e muitas outras. No entanto, estas aplicações não podem, como é óbvio, oferecer todo o conjunto de funcionalidades original, dada a abismal diferença de recursos existente.

Uma vez que o SIAG é uma aplicação web onde a parte cliente é executada num browser, é importante compreender as capacidades e limitações do browser disponibilizado no dispositivo móvel, neste caso o *PIE*[4, 13].

Após a análise das capacidades do *PIE* conseguiu-se perceber quais as suas limitações tecnológicas. Ao cruzar as limitações do *PIE* com as tecnologias utilizadas na interface cliente do SIAG obtiveram-se as seguintes conclusões:

1. HTML

   A versão *HTML* suportada pelo *browser* não reconhece alguns dos elementos presentes na versão mais recente, aquela em que a interface do SIAG se baseia. Deste modo, elementos fundamentais na aplicação como a *I-Frame* utilizada na interface original para alojar código *JavaScript* responsável por contactar o servidor e receber a resposta, evitando assim que a página visível ao utilizador seja recarregada durante estas interacções cliente-servidor, não podem ser utilizados na interface móvel. Outra característica importante desta versão é a ausência de alguns eventos nos elementos existentes, tais como o evento *onClick* num elemento *Image*.

2. CSS

   No caso do *CSS (Cascading Style Sheets)* a versão suportada pelo *PIE* também é bastante mais limitada que a de um browser tradicional. Estas limitações tornam impossível utilizar as mesmas etiquetas *CSS* para as duas interfaces, é necessário criar etiquetas próprias para a interface móvel.

3. Javascript

   O *PIE* anuncia que não suporta *JavaScript*, apenas *JScript*. O *JScript* é uma implementação da Microsoft do standard também implementado pelo *JavaScript*. Esta característica torna as duas implementações bastante idênticas, o que possibilita utilizar *JScript* no *Mobile* à imagem do *JavaScript* no SIAG, apesar de algumas limitações.

4. Java Applet

   O *PIE* não tem qualquer suporte para *Java Applets*, o que torna impossível a existência desta tecnologia nas interfaces móveis. Esta limitação teve um forte impacto no projecto, uma vez que as *Applets* são fundamentais na interface original e muito do código da camada de apresentação interage com estas. Numa primeira abordagem estas partes da camada de apresentação aparentavam ser impossíveis de utilizar pelo módulo *Mobile*.

Com estas limitações o servidor de mobilidade não pode ser um simples mecanismo que optimize as dimensões das páginas apresentadas no SIAG para o ecrã de um dispositivo móvel. É necessário implementar mecanismos que permitam gerar conteúdos num formato compatível e optimizado para os dispositivos.

Capítulo 4

Metodologia e Planeamento

No inicio do projecto foram estabelecidas quais as funcionalidades que deveriam ser desenvolvidas para o Servidor de Mobilidade. Este capítulo descreve a Metodologia de desenvolvimento e o seu impacto no Planeamento do projecto.

4.1 Metodologia de desenvolvimento

O processo de desenvolvimento de um projecto de software é bastante complexo e envolve diversas fases. Durante anos, diferentes visões sobre como estruturar todo este processo foram emergindo, algumas conseguiram uma elevada aceitação enquanto outras nunca se conseguiram impor. O principal objectivo que se pretende atingir é encontrar metodologias que melhorem a produtividade e qualidade do trabalho de desenvolvimento. A metodologia define um conjunto de práticas que podem ser executadas repetidamente para produzir o software.

As metodologias podem-se caracterizar como preditivas ou adaptativas[2]. Uma metodologia preditiva foca-se em planear o futuro em detalhe. Uma equipa que utilize esta metodologia consegue apresentar com exactidão que funcionalidade e tarefas estão planeadas para toda a extensão temporal do processo de desenvolvimento. Este detalhe e rigor torna difícil alterar o rumo do desenvolvimento, o plano definido no inicio está optimizado para a meta inicial e a mudança de rumo pode causar imensos danos no trabalho já desenvolvido.

No outro extremo estão as metodologias adaptativas, estas têm como objectivo uma rápida adaptação às mudanças de rumo necessárias no projecto. Quando os requisitos do projecto se alteram uma equipa adaptativa segue estas mudanças. Pode ser difícil à equipa descrever com exactidão o que irá acontecer no futuro, porém consegue enumerar com precisão que tarefas vão ser realizadas na semana seguinte, mas apenas que funcionalidades estão planeadas para o próximo mês.

4.1.1 Metodologia Ágil

A GEDI adopta no seu processo de desenvolvimento uma abordagem adaptativa, através do uso de uma Metodologia Ágil[3], mais concretamente são utilizadas metodologias *Extreme Programming* e *Feature Driven Development*.

As Metodologias Ágeis tentam minimizar os riscos através do desenvolvimento em pequenas caixas temporais denominadas iterações, estas duram tipicamente entre uma a quatro semanas. Cada iteração inclui todas as tarefas necessárias à produção de uma nova funcionalidade do produto. Estas tarefas incluem planeamento, análise de requisitos, desenho, codificação, testes e documentação. Uma vez que esta metodologia privilegia o contacto verbal em detrimento de documentos escritos a fase de documentação produz menos resultados. Esta abordagem pretende que no fim de cada iteração seja possível disponibilizar uma nova versão do produto onde já está incluida a funcionalidade implementada. Assim, é possível avaliar a evolução do sistema e definir novas prioridades.

Uma prática que também é utilizada no processo de desenvolvimento é a refactorização de código. Esta tarefa não corrige anomalias de funcionamento do produto nem adiciona novas funcionalidades, o seu propósito é melhorar a legibilidade do código, alterando a sua estrutura e removendo partes desnecessárias. Com isto torna-se mais fácil realizar a manutenção do código no futuro.

No fim de cada iteração são realizados testes unitários de forma a validar o código produzido. Após a integração da nova funcionalidade no produto já desenvolvido são feitos os testes de integração, de forma a verificar se o comportamento das funcionalidades anteriores se mantém válido.

4.2 Enquadramento Metodologia/Planeamento

A metodologia adoptada no processo de desenvolvimento influencia o planeamento do trabalho. Ao utilizar uma Metodologia Ágil, leva a que o planeamento não esteja definido em torno das fases de Análise, Desenho, Implementação e Testes. Como seria de esperar se fosse seguida uma abordagem preditiva. Desta forma, e uma vez que a Metodologia Ágil, define o processo de desenvolvimento em pequenas caixas temporais, o planeamento do projecto apresenta as diversas iterações a realizar.

O planeamento inicial definia intervalos temporais para a realização das tarefas. Esta abordagem, não sendo característica de uma metodologia adaptativa, foi necessária devido à duração do estágio. Desta forma, conseguiu-se planear o projecto para que no final a solução estivesse implementada.

4.3 Planeamento Detalhado

O planeamento detalhado é apresentado na tabela 4.3.

Planeamento Detalhado do Projecto Servidor de Mobilidade

Tarefa	Nome	Início	Fim	Duração	Realizado
1	Investigar Tema	12 Set	22 Set	9 dias	100%
2	Página Protótipo para Mobile	25 Set	29 Set	5 dias	100%
3	Implementação do Ecrã Autenticação	2 Out	5 Out	4 dias	
3.1	Detectar dispositivo móvel	2 Out	2 Out	1 dia	100%
3.2	Desenho Ecrã Autenticação	3 Out	3 Out	1 dia	100%
3.3	Operação Autenticação	4 Out	4 Out	1 dia	100%
4	Levantamento Interfaces existentes	6 Out	9 Out	2 dias	100%
5	Formação Framework	11 Out	11 Out	1 dia	100%
6	Cockpit SIAG dispositivo móvel	9 Out	25 Out	13 dias	
6.1	Desenho geral	9 Out	13 Out	5 dias	100%
6.2	Interface dinâmica de Menus	16 Out	18 Out	3 dias	100%
6.3	Zona de mensagens de erro	19 Out	19 Out	1 dia	100%
6.4	Navegação entre menus	20 Out	23 Out	2 dias	100%
6.5	Layout SIAG dispositivo móvel	24 Out	25 Out	2 dias	100%
7	Opção Preferências	26 Out	3 Nov	7 dias	100%
8	Protótipo de Ecrãs Tipo Lista	6 Nov	15 Nov	8 dias	
8.1	Desenho Lista	6 Nov	8 Nov	3 dias	100%
8.2	Visualizar registos	9 Nov	13 Nov	3 dias	100%
8.3	Navegação nas páginas da lista	14 Nov	15 Nov	2 dias	100%
9	Protótipo de Ecrãs Input	16 Nov	24 Nov	7 dias	
9.1	Desenho Ecrã Input	16 Nov	17 Nov	2 dias	100%
9.2	Operação Novo registo	20 Nov	22 Nov	3 dias	100%
9.3	Operação Alterar registo	23 Nov	24 Nov	2 dias	100%
10	Desenho da Componente de Criação de Opções Mobile (Interface Browser)	27 Nov	9 Mar	75 dias	
10.1	Estudo da ferramenta GEDIDev	27 Nov	1 Dez	5 dias	100%
10.2	Ecrãs Lista	4 Dez	15 Dez	10 dias	100%
10.3	Input	18 Dez	9 Mar	60 dias	
10.3.1	Campos	18 Dez	26 Jan	30 dias	
10.3.1.1	Input Standard	18 Dez	12 Jan	20 dias	100%

continua na página seguinte

Planeamento Detalhado do Projecto Servidor de Mobilidade (*continuação*)

Tarefa	Nome	Início	Fim	Duração	Realizado
10.3.1.2	Combobox	15 Jan	26 Jan	10 dias	100%
10.3.2	Zonas de Ecrã Tipo(Situações, Detalhe)	29 Jan	9 Mar	20 dias	100%
11	Integração das Componentes Desenhadas na Estrutura de Menus e Privilégios da Aplicação	12 Mar	6 Abr	20 dias	100%
12	Pesquisa e Ordenação de registos	9 Abr	4 Mai	20 dias	
12.1	Desenho Ecrã Pesquisa	9 Abr	13 Abr	5 dias	100%
12.2	Desenho Ecrã Ordenação	16 Abr	20 Abr	5 dias	100%
12.3	Filtros de Pesquisa e de Ordenação	23 Abr	27 Abr	5 dias	100%
12.4	Lista de resultados e persistência do conjunto	30 Abr	4 Mai	5 dias	100%
13	Desenho da Componente de Criação de Opções Mobile (Interface Móvel)	7 Mai	1 Jun	20 dias	100%
14	Relatório Final	4 Jun	22 Jun	15 dias	100%

Planeamento Detalhado do Projecto Servidor de Mobilidade

4.4 Mapa de Gantt

Com a ajuda de um mapa de Gantt consegue-se perceber a verdadeira extensão de cada tarefa, com a respectiva duração e dependências. O mapa de Gantt está representado nas figuras 4.1, 4.2, 4.3 e 4.4.

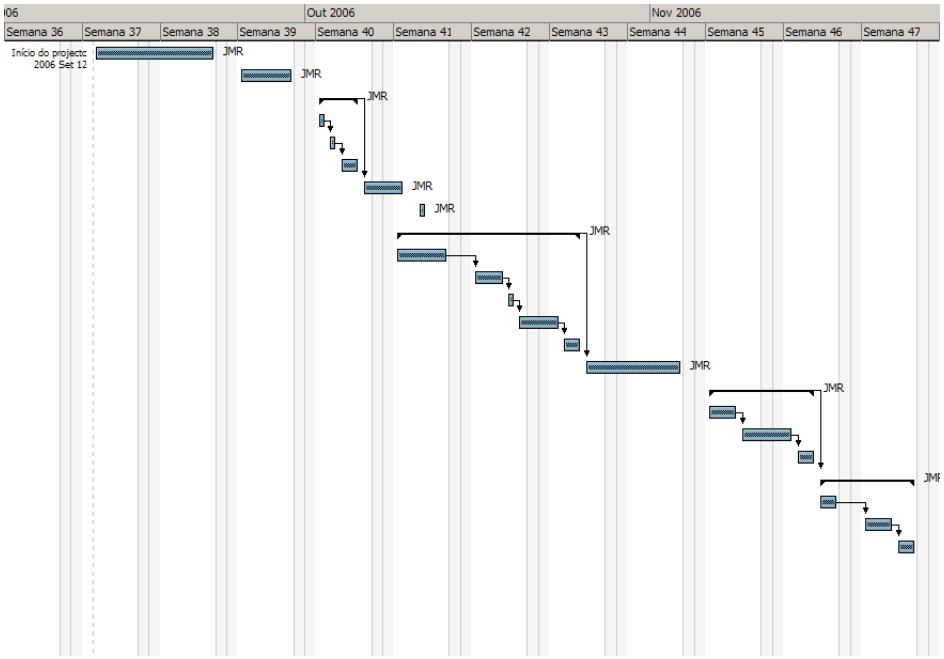

Figura 4.1: Mapa de Gantt - parte 1

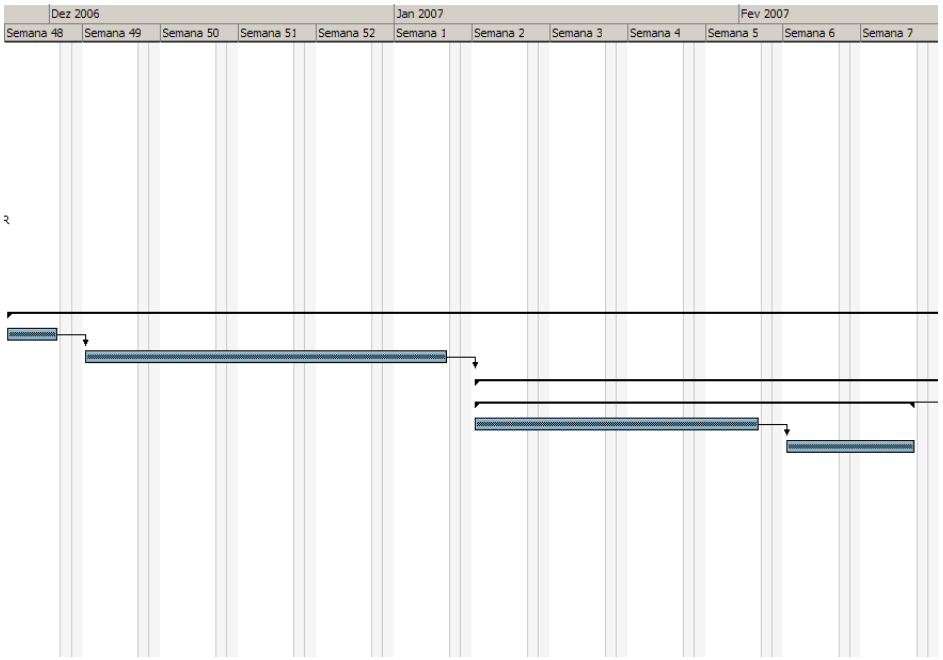

Figura 4.2: Mapa de Gantt - parte 2

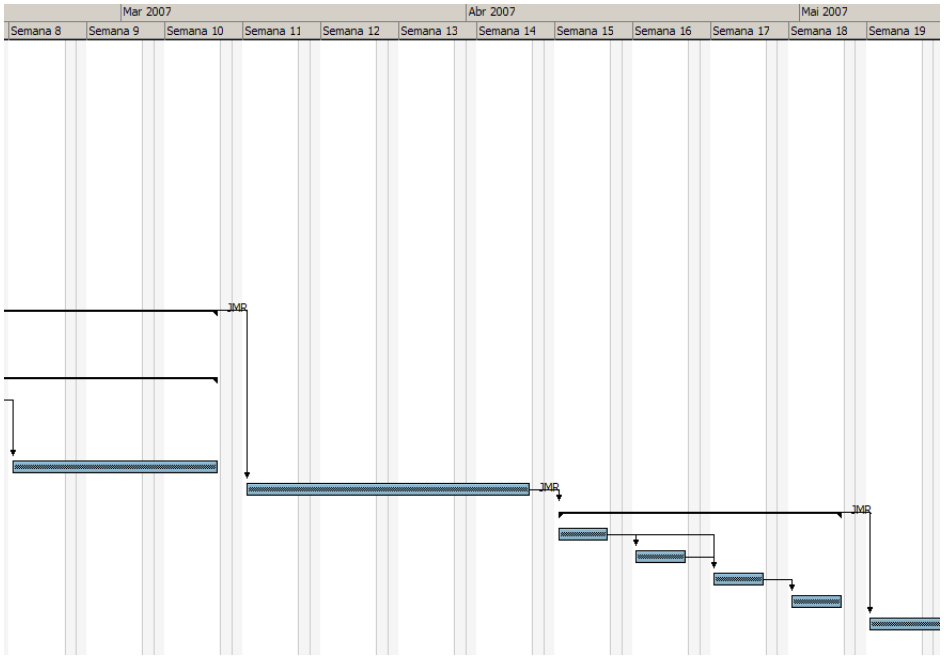

Figura 4.3: Mapa de Gantt - parte 3

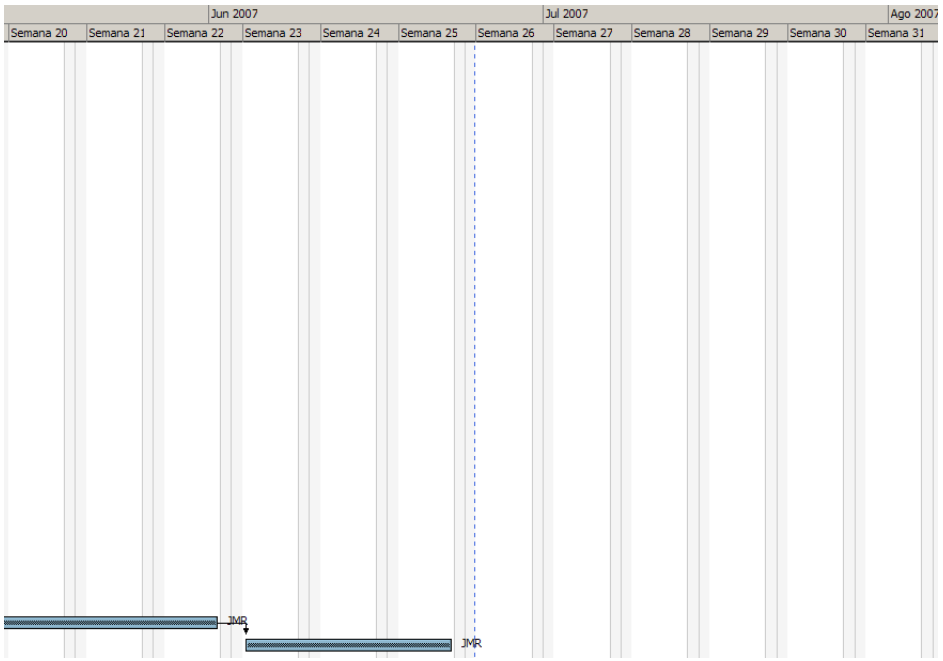

Figura 4.4: Mapa de Gantt - parte 4

Capítulo 5

Trabalho realizado

Neste capítulo é abordado o trabalho realizado no processo de desenvolvimento do servidor de mobilidade. Em complemento a este assunto é descrito o conjunto de ferramentas utilizadas no desenvolvimento da solução.

5.1 Ferramentas utilizadas

A produção de uma solução informática requer a preparação de um ambiente de desenvolvimento adaptado às tecnologias utilizadas na solução. Estando este projecto relacionado com uma aplicação web é igualmente necessário configurar um ambiente de execução adequado, isto é, instalar um servidor de aplicações e um SGBD.

O Ambiente de Desenvolvimento Integrado utilizado neste projecto foi o MyEclipse, este está vocacionado para o desenvolvimento de aplicações web que utilizem a linguagem *Java*. Outra característica importante do MyEclipse é a capacidade de integração com um vasto conjunto de servidores de aplicações e SGBDs. Esta capacidade agiliza o processo de desenvolvimento, facilitando muitas tarefas tais como, controlar o servidor e o estado das aplicações implantadas no mesmo, a implantação automática e em tempo real das alterações ao código da aplicação que estamos a desenvolver. Ao nível da integração com o SGBD disponibiliza uma interface para uma fácil interacção, permitindo realizar várias operações sobre o SGBD. Outro aspecto importante do MyEcplise é permitir utilizar uma ferramenta de controlo de versões, neste caso o *CVS (Concurrent Versions System)*.

O servidor de aplicações instalado foi o Apache Tomcat. Como este projecto envolve a tecnologia *JSP* é necessário utilizar um servidor que suporte *servlets* e *JSP*, ou seja, um servidor que disponibilize um ambiente onde o código *Java* possa ser executado em colaboração com um servidor web. Tudo isto está implementado no Apache Tomcat.

O SGDB utilizado foi o MySQL Community Server[9], as principais características deste SGBD é o facto de ser de uso gratuito e poder ser executado em várias plata-

formas.

5.2 Etapas do desenvolvimento

5.2.1 Detectar dispositivos móveis

Após a introdução ao tema do projecto iniciou-se a exploração da tecnologia *JSP* e a criação de algumas páginas com conteúdos dinâmicos. A primeira tarefa do projecto foi desenhar uma página de autenticação móvel e desenvolver um mecanismo que permita avaliar se o pedido da página inicial (*index.jsp*) foi originado por um dispositivo móvel ou por um *browser* tradicional. Na posse desta informação a resposta do servidor é direccionada para a página de autenticação correspondente.

A solução encontrada foi codificada na página *index.jsp*. Dentro de uma etiqueta *JSP* adicionou-se código *Java* que analisa o cabeçalho do pedido *HTTP (HyperText Transfer Protocol)*, este código avalia o tipo de *browser* cliente, consoante o resultado da avaliação encaminha o pedido para a página de autenticação correcta.

5.2.2 Operação de Autenticação Móvel

Depois de implementada a capacidade de detecção do tipo de pedido que chega ao servidor tratou-se da operação de autenticação. Os mecanismos de autenticação utilizados no *Mobile* são os mesmos da Base, as alterações implementadas para atingir esta solução ocorreram no ficheiro *struts-config.xml* na definição da entrada da *Action Login* e na *ActionClass* correspondente. Na *Action Login* foram adicionados dois novos pontos de encaminhamento para páginas *Mobile* enquanto que na *ActionClass* foi alterado o código que retorna o nome da entrada de encaminhamento que o *Struts* deve utilizar.

5.2.3 Cockpit Mobile e Layout Mobile

A interface é constituida por duas componentes o *Cockpit* e o *Layout*. O *Cockpit Mobile* é a estrutura de interface que apresenta os menus, a árvore de navegação, as mensagens de erro, botões de sair e de informação sobre o utilizador, e a versão da aplicação. O *Layout Mobile* complementa o *Cockpit*, é um nível de apresentação inserido dentro deste. Este nível é o esqueleto de todos os ecrãs que apresentam formulários ou listas de registos, é aqui que é colocado o título do ecrã e os botões consoante o tipo de operação (validar, cancelar). O *Cockpit Mobile* e o *Layout Mobile* podem ser observados nas figuras 5.1 e 5.2 respectivamente.

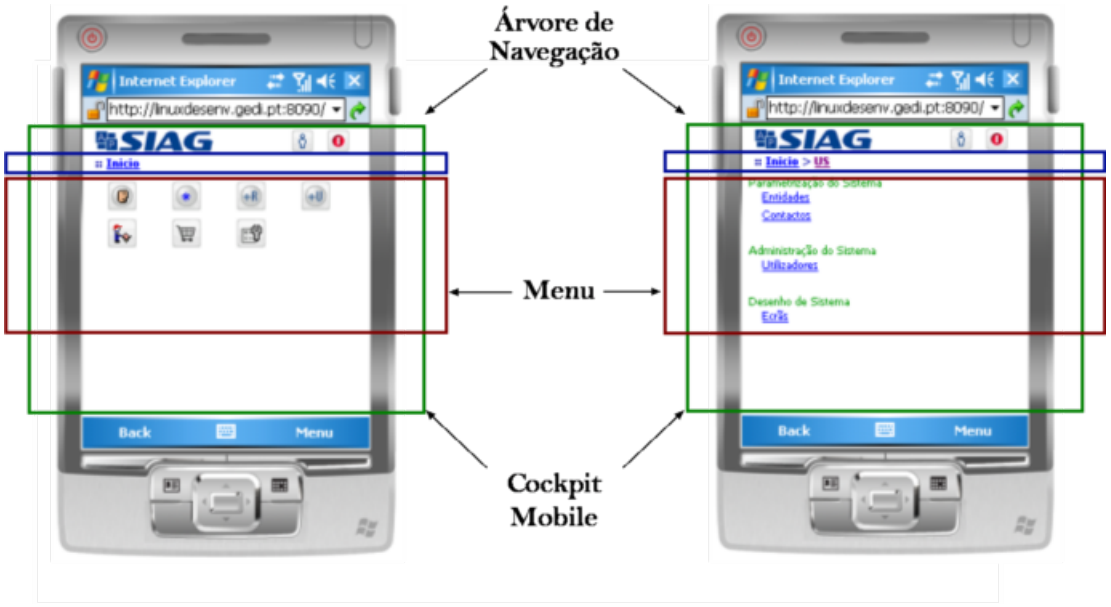

Figura 5.1: Cockpit Mobile

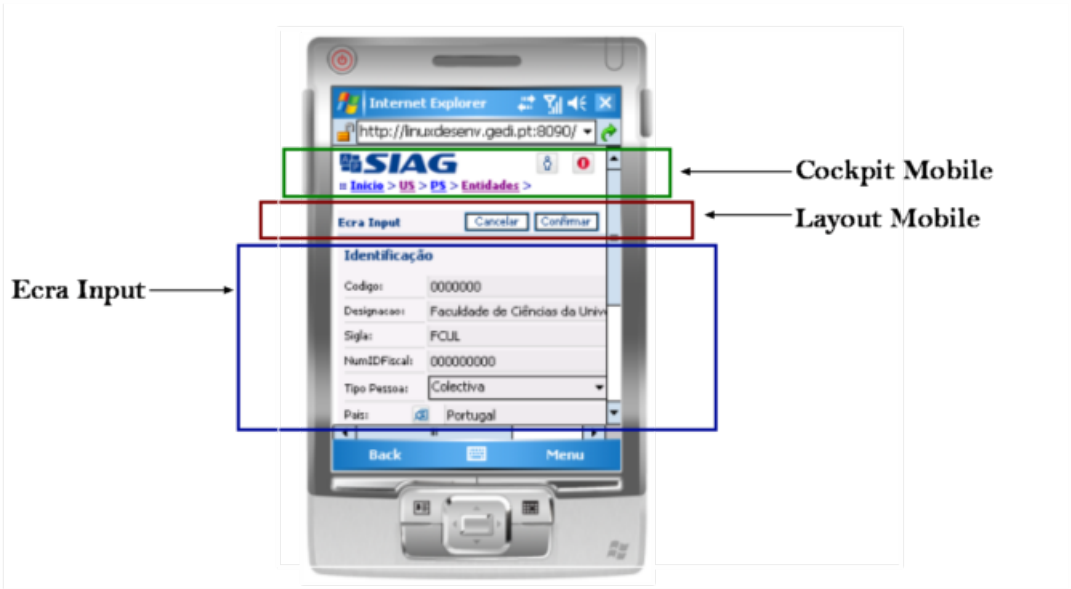

Figura 5.2: Layout Mobile

Interface Dinâmica de Menus

A estrutura dos menus da aplicação obedece a uma divisão em Módulos, que contêm Áreas, que por sua vez englobam Componentes que aceitam Operações. Hierarquia dos menus:

- ☞ Módulo
- ☞ Área
- ☞ Componente
- ☞ Operação

Na figura 5.1 está um exemplo da apresentação dos Módulos da aplicação e na figura 5.2 estão representados as Áreas de um Módulo e respectivas Componentes.

O sistema após uma operação de autenticação bem sucedida, carrega uma colecção de componentes e respectivas operações para as quais o utilizador tem privilégios. Ao iterar esta colecção são filtrados os Módulos que não conduzem a nenhuma componente para a qual o utilizador tenha privilégios. Desta forma optimiza-se o espaço para a apresentação e navegação do menu.

Este foi o primeiro critério a ser aplicado na construção do menu.

5.2.4 MobBaseAction

A tecnologia *Struts* utiliza para cada *Action* definida no ficheiro de configuração uma *ActionClass* e um *ActionForm*.

A *ActionClass* é onde se trata a lógica de cada *Action*, pertence à camada de Apresentação e interage com a camada de Negócio através de serviços. Cada *ActionClass* tem designado um serviço que é responsável por tratar da sua lógica de negócio.

Os *ActionForms* são objectos *Java* que mapeiam um atributo para cada campo do formulário *HTML*. A apresentação é feita através de *JSP*. Os dados que são apresentados na *JSP* são obtidos através de *ActionForms* que são construídos na componente de controlo e que são mapeados para a *JSP* de forma automática pelo *Struts*.

Devido às limitações tecnológicas do *PIE* não se consegue tirar o máximo partido da estrutura de apresentação já desenvolvida na *Framework* GEDI. Não é possível utilizar *ActionForms* uma vez que os campos apresentados no ecrã original não são todos transpostos para a interface *Mobile*. Logo, o mapeamento feito pelo *ActionForm* deixa de estar consistente e o sistema não consegue superar esta situação.

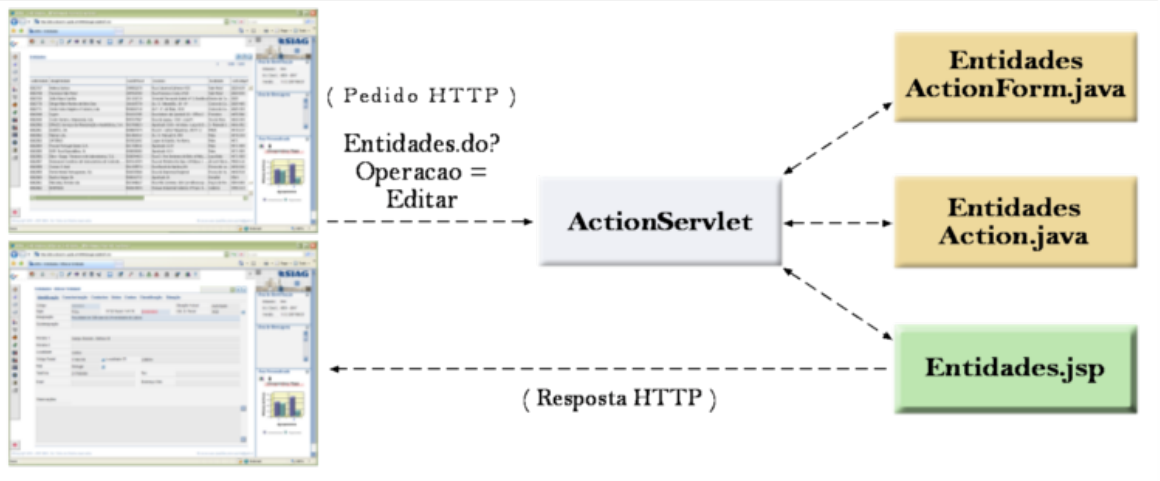

Figura 5.3: Interacção Struts

Cada *Action* definida no *Struts* teria de ser duplicada para suportar as limitações existentes no *Mobile*, esta abordagem, por originar um aumento de complexidade e redundância, teve de ser relegada.

Na solução proposta para resolver este problema apenas uma nova *Action* tem de ser adicionada ao *Struts*. Todos os pedidos que cheguem ao servidor provenientes de um dispositivo móvel apontam para esta entrada (*MobileGenericAction.do*) e a *ActionClass* indicada na configuração processa o pedido e encaminha-o para a apresentação correcta.

Esta abordagem genérica estende-se também às páginas de apresentação de listas de registos e de formulários. Ou seja, existe apenas uma página *JSP* para apresentar as listas e outra para apresentar formulários.

Na figura 5.3 está representada uma interacção entre um *browser* tradicional e o servidor. O pedido feito pelo *browser* indica qual a *Action* à qual este é direccionado e qual a operação. O *Struts* invoca então a *ActionClass*, o *ActionForm* e a *JSP* associados a esta *Action*, o pedido é processado e é enviada uma resposta ao *browser*. O mecanismo implementado no módulo *Mobile* pode ser observado na figura 5.4, aqui o pedido é feito para a *Action* genérica, indicando tal como no interacção anterior qual a operação a realizar e, além disso, qual a componente alvo desta operação. O *Struts* invoca a *ActionClass* associada à *Action* genérica, não existe nenhuma referência a um *ActionForm* e a página de resposta é gerada recorrendo a uma *JSP* genérica.

5.2.5 Ecrãs do Tipo Lista

Aqui surgiu o primeiro grande desafio. A apresentação de listas de registos na interface original recorre a *Java Applets*, no *PIE* não é possível implementar esta

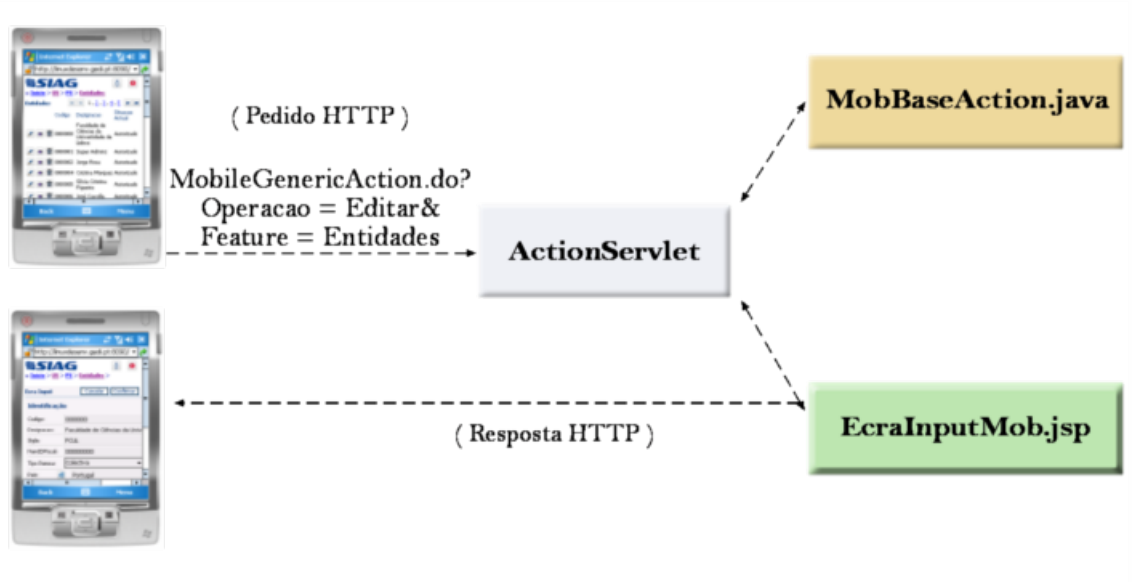

Figura 5.4: Interacção Struts Mobile

solução uma vez que esta tecnologia não é suportada pelo *browser*. A alternativa na interface *Mobile* é apresentar os registos através de uma tabela *HTML* e implementar mecanismos que permitam suportar o elevado número de entradas a apresentar neste tipo de ecrã. A *JSP* responsável por esta apresentação é totalmente independente da componente cujos registos são exibidos.

O projecto Base disponibiliza uma interface onde é possível para cada componente definir um conjunto de vistas para a apresentação dos registos. Cada vista representa uma configuração das colunas que devem ser exibidas no ecrã. Estas vistas estão associadas ao utilizador que as criou, mas podem ser partilhadas para todos os outros, tornando-as assim globais. O utilizador para cada componente tem definida a sua vista por omissão, caso nunca a tenha definido a vista é vazia.

Esta funcionalidade foi aproveitada para permitir configurar vistas para a interface *Mobile*. Foram no entanto definidos novos critérios para as vistas de omissão do utilizador em cada componente.

Critérios para a escolha da vista por omissão da componente para a interface *Mobile*:

- ☞ Vista criada pelo utilizador com designação “Mobile”
- ☞ Vista com designação “Mobile”e partilhada para Todos
- ☞ Vista por omissão do utilizador na interface original

Se nenhum destes critérios for satisfeito a interface *Mobile* não apresentará a opção de consultar lista de registos para a componente em causa.

Implementação

A configuração da vista é guardada numa estrutura *XML*, quando o servidor recebe um pedido para listar os registos de uma componente utiliza o código desta e a informação sobre o utilizador para decidir qual a estrutura *XML* que deve consultar. Na posse da informação contida no *XML* pode-se definir as colunas a apresentar na tabela. Os registos são carregados do SGBD utilizando métodos já existentes na *Framework* GEDI é então filtrado o conjunto que deve ser exibido, pois existe um número máximo de entradas por cada página no ecrã.

A informação de quais as colunas a apresentar e o conjunto de registos da página são passados à *JSP* que desenha a tabela e escolhe quais os campos dos registos que devem ser exibidos de acordo com a configuração da vista.

Após ter sido implementada a *JSP* responsável por gerar o ecrã de listas a partir de uma estrutura de dados, teve de se estudar qual a melhor forma de navegar nas páginas de registos da lista. Isto porque, se o utilizador pedir para listar uma componente com um milhar de registos, não é praticável carregar todos os registos e exibi-los no ecrã de uma só vez. A opção tomada foi definir um número máximo de registos por página e criar uma estrutura de navegação com ligações para o utilizador pedir novas páginas.

5.2.6 Ecrã do Tipo Input

O ecrã de tipo Input é utilizado para apresentar todo o tipo de formulário que represente um registo de uma qualquer componente. Na interface original, cada componente possui o seu ecrã de formulário e uma *Action* definida no ficheiro de configuração do *Struts*, nesta entrada está indicada a *ActionClass* e o *ActionForm* que tratam dos pedidos da componente.

O *Struts* quando recebe um formulário *HTML* vindo do *browser* preenche automaticamente o *ActionForm* com os valores dos campos do formulário. A partir deste *ActionForm* é preenchido um objecto que designamos por *pojo*, trata-se de um objecto *Java* que mapeia todos os campos e relações da tabela da base de dados que corresponde à componente. O *pojo*, após este procedimento, contém todas as alterações feitas no ecrã pelo utilizador, este depois de validado é persistido na base de dados.

Este mecanismo repete-se para todos os ecrãs da aplicação. No entanto, esta abordagem não pode ser seguida no *Mobile*, uma vez que existe apenas uma *Action* definida no *Struts* e não existe *ActionForm*. Para ultrapassar esta limitação foi implementado um mecanismo que consegue reflectir as alterações do formulário no *pojo* de ecrã. Este mecanismo entra em acção a partir do momento que o servidor recebe o pedido de criar/editar/consultar um registo, este pedido vai resultar na

apresentação de um ecrã de input.

O processo necessário a construir a página do ecrã de input implica que se obtenha uma lista com os campos que devem ser apresentados e suas características. Esta informação está presente numa estrutura *XML* que guarda uma configuração de ecrã. Ao iterar todos os campos é construída uma estrutura que representa uma colecção de configurações de cada um dos campos. A configuração base de um campo é apresentada na tabela 5.1, consoante o tipo surgem outras propriedades necessárias.

–	Propriedades
1	Tipo (ex: Tabulador, Arealist, Checkbox, Combobox, Texto...)
2	Nome do elemento HTML
3	Label do campo
4	Formato de input (ex: Integer, Timestamp, String...)
5	Obrigatório.
6	Apenas de leitura

Tabela 5.1: Configuração Base de um Campo

Para que os valores sejam enviados para o servidor no acto de submeter o formulário todos os campos aí representados são elementos *HTML* do tipo input. Em consequência desta operação, o servidor recebe como parâmetro para cada um destes campos um par do tipo <*id do elemento, valor do campo*>. O conjunto destas propriedades possui toda a informação necessária para construir um elemento *HTML* que represente o campo no formulário.

O identificador do elemento input de cada campo é especificado na estrutura que é passada à *JSP* para construir a página e obedece a algumas regras. Caso o campo represente uma coluna da tabela da componente, o identificador é o próprio nome da coluna, se corresponder a uma coluna de uma tabela relacionada com a tabela da componente, o identificador representa todo o caminho até essa coluna (ex: TComponente.TRelacionada.Coluna).

A operação de envio de um formulário *Mobile* tem como *Action* destino a **MobileGenericAction**. O *Struts* quando recebe uma operação deste tipo delega o controlo de processar o formulário na *ActionClass* (**MobileBaseAction**) que trata os pedidos feitos a esta *Action*. A *ActionClass* carrega novamente a configuração do ecrã a partir de *XML* e obtém assim a lista de campos presentes no formulário recebido. Para cada elemento obtido na iteração da lista, normaliza-se o seu nome para que este corresponda ao identificador do elemento presente no formulário. Avalia-se o parâmetro que o servidor recebeu com chave igual ao nome normalizado e obtêm-se assim o valor do campo do formulário, actualiza-se o valor do campo no *pojo* de

ecrã. Quando se tiver terminado este processo então é feita a validação do *pojo* e persistido na base de dados caso tudo esteja correcto.

5.2.7 Zonas Tipo Listas

Os ecrãs da interface Base possuem zonas nas quais são apresentadas Listas que se convencionou designar por *Arealists*. Cada *Arealist* é uma *Applet* carregada no *browser* que comunica directamente com o servidor para efectuar operações neste. É utilizada uma *JTable* para listar o conteúdo da *Arealist*. A configuração das colunas está guardada num ficheiro *XML* único para cada *Arealist*. Todos os dados apresentados por esta são guardados no servidor num objecto denomidado *AppletContext*, este objecto é instanciado quando é invocada a operação de mostrar todos os elementos da lista. Consoante o tipo da *Arealist* exibida é possível adicionar, remover e editar linhas. Estas operações comunicam directamente com o servidor actualizando a instância do *AppletContext*. Para cada tipo de *Arealist* existe uma *Action* definida no *Struts* que indica qual a *ActionClass* que processa as operações.

Estas zonas de ecrã, para serem apresentadas na interface *Mobile*, têm de recorrer a outra forma de apresentação uma vez que não existe suporte para *Applets* no *browser* dos dispositivos móveis.

No *Mobile* a solução para representar as *Arealists* passa por especificar no ficheiro *XML* de configuração quais as colunas obrigatórias para esta interface. As linhas de *Arealist* são apresentadas numa tabela *HTML* sendo possível visualizar o detalhe de cada uma. Esta operação de detalhe utiliza um ecrã do tipo input para exibir a informação de todas as colunas da linha. A capacidade de editar um campo respeita as propriedades da linha e do próprio campo. A interacção directa entre as *Arealists* e o servidor causa problemas na interface *Mobile*, isto porque de cada vez que ocorre um pedido da *Arealist* ao servidor este responde com uma nova página para actualizar o estado da lista. As alterações feitas no formulário são perdidas, é pois necessário implementar um mecanismo que consiga registar as alterações e enviá-las ao servidor juntamente com o pedido da *Arealist*. No lado do servidor, antes do processamento do pedido, são actualizados os objectos que estão representados no ecrã, ou seja, o *pojo* de ecrã da componente e os *AppletContext* das *Arealist* exibidas. Esta solução recorre a *JScript* no lado do *browser*, todos os elementos input têm definido no evento *onChange* a função que deve ser executada para registar o novo valor do campo, nos botões da *Arealist* (adicionar, remover, detalhe) é definido no evento *onSelect* uma função que junta todas as alterações aos parâmetros passados ao servidor e envia o pedido da operação.

A cada *Arealist* está atribuída uma *Action* no ficheiro de configuração do *Struts*, a *ActionClass* correspondente é responsável por processar as operações que a *Arealist* invoca no servidor. Na interface *Mobile* todas as operações invocadas no ecrã têm

como destino a *Action* genérica desta interface. A *ActionClass* que trata os pedidos dirigidos a esta *Action* não consegue implementar as diferentes especificidades de cada *Arealist*. De forma a resolver este problema foi criada uma estrutura onde estão listadas todas as componentes, para cada uma destas quais as *Arealists* que existem no ecrã original, e por fim para cada *Arealist* a informação sobre o nome do ficheiro *XML* com a configuração das colunas e qual a classe serviço da *Arealist*. Outra funcionalidade implementada permite obter a *ActionClass* de uma *Arealist* através do seu nome. Assim, quando a *ActionClass* da *Action* genérica é invocada para tratar operações de *Arealists*, a mesma prepara uma estrutura que simula um pedido vindo directamente do ecrã. Utilizando o mecanismo de reflexão disponibilizado pelo *Java* a operação pedida pela *Arealist* é executada na sua *ActionClass* e no seu serviço passando a estrutura correctamente inicializada. A operação é assim executada na classe correcta, utilizando as implementações já existentes, a ***ActionClass Mobile*** funciona apenas como um adaptador.

5.2.8 Pesquisa e Ordenação

O projecto Base disponibiliza as funcionalidades de pesquisa e de ordenação no ecrã que lista os registos de uma componente. Estas duas funcionalidades são essenciais para se conseguir restringir/optimizar o conjunto de resultados apresentados, ainda mais quando estamos a trabalhar com dispositivos com interfaces de dimensão tão limitada como é o caso dos dispositivos móveis.

Foram desenhado dois novos ecrãs para a interface *Mobile* destas duas funcionalidades. Cada um destes ecrã possui uma *Arealist*, num a de Pesquisa e no outro a de Ordenação. Estas *Arealists* permitem adicionar linhas com critérios de pesquisa ou de ordenação consoante o ecrã, os critérios podem ser aplicados sobre qualquer campo existente no ecrã da componente sobre a qual se está a fazer a pesquisa/ordenação, mesmo que este campo pertença a uma outra tabela relacionada com a tabela da componente.

O funcionamento destes dois ecrãs é semelhante ao de outro ecrã do tipo input, a principal diferença é o facto de a ***ActionClass Mobile*** reconhecer as operações vindas destes dois ecrãs e implementar métodos optimizados para as processar. Esta particularidade deve-se à necessidade de controlar as listas de tabelas e respectivas colunas que preenchem as *combobox* de cada linha. As *Arealists* destes ecrãs são processadas utilizando o mesmo modelo que todas as outras, a execução da lógica é delegada na *ActionClass* e no serviço respectivo.

Os resultados obtidos após a submissão destes critérios são processados da mesma forma que qualquer outro conjunto de registos proveniente de uma consulta normal. Este conjunto é depois passado ao ecrã de listas e é gerada a apresentação.

5.2.9 Ecrã de Desenho de Ecrãs

A configuração dos campos apresentados no ecrã de input de cada componente é dinâmica. Para carregar o ecrã estas configurações são lidas de uma estrutura *XML*. Esta funcionalidade de desenhar ecrãs foi implementada especialmente para o servidor de mobilidade, mas possui potencialidades para ser utilizado em outros projectos.

Na interface criada para desenhar ecrãs selecciona-se a componente para a qual estamos a desenhar, consoante a componente escolhida é carregada a informação sobre todas as tabelas da base de dados com a qual a tabela da componente está relacionada. Esta lista vai preencher uma *combobox* de tabelas, ao seleccionar uma tabela na *combobox* são carregados os campos desta tabela. Estes são apresentados numa *Arealist* não editável (*colunasArealist*), é esta que contém a configuração do ecrã que estamos a desenhar. As colunas apresentadas nesta *Arealist* são:

- ☞ Objecto
- ☞ Tabela
- ☞ Campo
- ☞ Titulo
- ☞ XML
- ☞ Caminho

Na coluna objecto existe uma *combobox* onde se escolhe o tipo do campo a apresentar no ecrã, se for um campo que tenha sido adicionado a partir da *Arealist* da esquerda as opções disponíveis são texto, *checkbox*, *combobox* e auxiliar. Os campos adicionados desta forma representam elementos de uma tabela no SGBD, se o campo for adicionado sem nenhuma relação com a *Arealist* campos representa uma linha vazia, neste caso a *combobox* Objecto apresenta as opções Tabulador e Lista. Um Tabulador representa uma divisão no ecrã, enquanto a Lista representa uma *Arealist*.

A coluna Tabela e a coluna Campo não são editáveis e apresentam o nome da tabela e do campo no SGBD à qual esta linha se refere. Na coluna título indica-se qual o título que deve ser apresentado no ecrã para o campo.

A coluna *XML* contém uma *combobox* cujas opções são preenchidas em duas circunstâncias, quando a opção seleccionada na coluna objecto indica *combobox* ou lista. Se o objecto for uma *combobox* na coluna *XML* são apresentadas todas as listas que preenchem as opções das *combobox* do ecrã original da componente. O nome das listas está indicado na *ActionClass* da componente. Se o objecto for

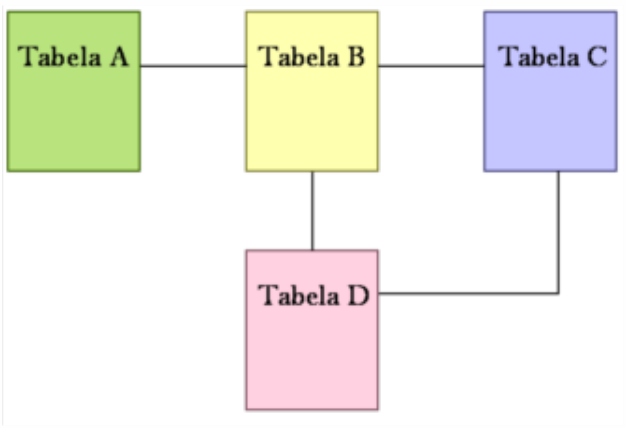

Figura 5.5: Caminhos entre tabelas

uma lista então são apresentadas todas as *Arealists* que existem no ecrã original da componente. A informação sobre a colecção de *Arealists* de um ecrã está presente do ficheiro **ResourcesAreaList**.

Na coluna caminho é apresentada uma *combobox* que é preenchida quando o campo associado à linha da *Arealist* pertence a uma tabela relacionada com a tabela principal da componente. As opções apresentadas na *combobox* são os caminhos possíveis para chegar a este campo navegando a partir da tabela componente. Por exemplo, para chegar à tabela D a partir da tabela A pode existir mais que um caminho, como se vê na figura 5.5. Todas estas hipóteses são apresentadas na configuração de ecrã.

Na base de dados foi criada uma tabela de nome Ecra para guardar as várias configurações, nesta tabela existe uma coluna onde está o código da componente a que a configuração se refere. A tabela Ecra tem também uma relação com a tabela *XML*, é nesta tabela que é guardada a estrutura *XML* com a configuração do ecrã.

Quando a *ActionClass* genérica de *Mobile* recebe um pedido para exibir o ecrã de uma componente, pede a configuração de ecrã à tabela Ecra e navega até à estrutura *XML*.

5.2.10 Interface Dinâmica

Nesta fase do projecto e depois de já implementados os mecanismos que possibilitam configurar as interfaces de Ecrã Lista e de Ecrã Input, é possível refinar os critérios de desenho da interface de menus.

Até este momento as opções presentes no menu *Mobile* reflectiam apenas os privilégios do utilizador no sistema. Esta abordagem leva a situações onde é apresentada a opção de navegar para uma componente para a qual não é possível realizar nenhuma operação, pois não existe nenhuma vista definida para o Ecrã de Lista, nem uma configuração para apresentar um Ecrã de Input para os registos desta compo-

nente. Para evitar esta situação foram adicionados novos critérios. No momento da operação de autenticação válida, e após obter a lista de privilégios, é executado uma rotina que avalia para todas as operações de cada componente presentes nesta lista, se existe uma configuração de vista e/ou de ecrã de input válidas para o utilizador. Ou seja, se para uma determinada componente o utilizador não tiver privilégios para criar, editar e consultar, não é necessário avaliar se existe uma configuração de ecrã para esta componente. Uma componente para a qual foi definida uma configuração de ecrã, mas não tenha uma definição de vista associada, apenas terá disponível na interface a operação de Novo.

As componentes que não satisfaçam nenhum dos dois requisitos (configuração de vista e configuração de ecrã) serão excluídas da interface de menus.

5.2.11 Ecrã Desenho de Ecrãs Mobile

A capacidade de desenhar ecrãs na interface móvel é uma funcionalidade importante para dotar a aplicação de uma maior independência e mobilidade. A aplicação deve conseguir adaptar-se às necessidades do utilizador.

A interface do ecrã de desenho foi implementada especialmente para esta funcionalidade, tal como já tinha acontecido com os ecrãs de Pesquisa e Ordenação. Esta opção deve-se à complexidade dos ecrãs, pois existem duas *Arealists* que interagem entre si.

De forma a conseguir apresentar as duas *Arealists* no espaço de uma só, as colunas Tabela e Campo, passam a ser editáveis e contêm uma *combobox* que é preenchida com as Tabelas e com os respectivos campos. Ao seleccionar uma nova Tabela numa das linhas, é necessário fazer um pedido ao servidor para preencher a *combobox* Campo dessa linha com os campos da tabela, tendo em atenção que todas as outras linhas devem se manter inalteradas e continuar a apresentar as *combobox* preenchidas com as tabelas e os campos correctos. Na interface *browser* não era necessário ter estas cautelas uma vez que estas colunas era campos de texto não editáveis e estas interacções de tabelas e campos estão isoladas noutra *Arealist* independente.

Capítulo 6

Conclusão

O processo de desenvolvimento de uma solução informática envolve bastantes riscos e desafios. Ao longo do estágio por diversas vezes me deparei com esta realidade.

Este projecto tinha como principal desafio as limitações apresentadas pelos dispositivos móveis. Desenvolver mecanismos que permitam extender a arquitectura inicial pensada para funcionar sem este tipo de limitações, e manter a independência entre projectos por vezes tornou-se difícil. Em alguns casos pontuais foi necessário efectuar alterações para que a *framework* tivesse conhecimento sobre qual o contexto em que estava a executar.

A falta de experiência no envolvimento com um projecto desta dimensão revelou-se em alguns momentos deste período. O fraco conhecimento da framework e da visão global da arquitectura no início do estágio, originou que o levantamento de requisitos por vezes fosse incorrecto. Estas situações tornam-se visíveis muitas das vezes apenas na fase de codificação, levando a que todo o processo tenha de ser reformulado. O facto de se utilizar uma Metodologia Ágil representa uma mais valia nestes casos, pois a adaptação às mudanças não afecta o trabalho já realizado. Ao focar a análise, desenho e implementação numa componente de cada vez, possibilita que o processo de implementação não tenha de obedecer a um conjunto de regras rígidas definidas na fase de análise/desenho realizada no inicio do projecto. A redefinição de uma destas regras iria afectar todo o trabalho já realizado. O uso de frameworks complexas que disponibilizam pouca documentação, é outro factor que por vezes dificulta a execução das tarefas.

O planeamento definido inicialmente foi cumprido e todas as funcionalidades foram implementadas e testadas.

Este projecto proporcionou-me um contacto directo com o processo de desenvolvimento de uma solução informática. O conhecimento adquirido ao longo do percurso académico foi posto à prova neste período. O resultado final é a consolidação de todo este conhecimento anteriormente adquirido e a abertura a uma nova realidade.

Acrónimos

CRM Customer Relationship Management, 41
CSS Cascading Style Sheets, 41
CVS Concurrent Versions System, 41

ERP Enterprise Resource Planning, 41

GEDI Gabinete de Estudos e Divulgação Informática, 41

HTML Hypertext Markup Language, 41
HTTP HyperText Transfer Protocol, 41

IDE Integrated Development Environment, 41

JSP Java Server Pages, 41

MVC Modelo-Vista-Controlo, 41

PDA Personal Digital Assistant, 41
PIE Pocket Internet Explorer, 41
POJO Plain Old Java Object, 41

SGBD Sistema de Gestão de Base de Dados, 41
SIAG Sistema Integrado de Apoio à Gestão, 41
SQL Structured Query Language, 41

XML Extensible Markup Language, 41

Glossário

Action Action é uma entrada no ficheiro de configuração do Struts., 42

ActionClass Classe associada a uma Action, responsável por processar o pedido e interagir com o código da camada de Negócio., 42

ActionForm Classe que mapeia todos os campos do ecrã, está associado a uma Action., 42

Applet Código Java executado num browser., 42

AppletContext Objecto responsável por guardar o estado das Arealist no servidor., 42

Arealist Designação atribuída às listas presentes nos ecrãs do SIAG implementadas através de uma applet., 42

Base Nome do projecto onde está implementada a Framework GEDI., 42

Browser Programa utilizado para visualizar documentos HTML., 42

CRM Termo que define toda uma classe de ferramentas utilizadas pelas empresas para gerir a sua relação com os clientes., 42

CVS Um sistema de controlo de versões. Mantém registo das alterações num conjunto de ficheiros., 42

ERP Define uma solução informática desenhada para integrar todos os dados e processos de uma organização., 42

Framework Estrutura conceptual utilizada para resolver problemas complexos. As soluções são desenvolvidas em torno da framework., 42

GEDI Instituição de acolhimento onde decorreu o estágio. Empresa cuja actividade principal é a concepção, desenvolvimento e implementação de soluções informáticas., 42

Hibernate Solução desenvolvida em Java que permite fazer um mapeamento objecto-relacional. O modelo relacional das tabelas da base de dados é mapeado em objectos Java., 42

HTTP Protocolo de comunicação utilizado para transferir dados na Internet., 42

IDE É uma aplicação que agrega as diferentes ferramentas necessárias ao desenvolvimento de software., 42

Java Linguagem de programação que segue a metodologia object-oriented (OO). A sua principal característica é a independência em relação à plataforma em que é executada., 42

JavaScript Linguagem de script executada num browser. Implementação do standard feita pela Netscape/Mozilla., 42

JScript Linguagem de script executada num browser. Implementação do standard feita pela Microsoft., 42

JSP Tecnologia Java que permite gerar conteúdo HTML dinamicamente., 42

Mobile Nome do módulo desenvolvido no âmbito do estágio., 42

Object-Oriented Paradigma da programação que se baseia no conceito de objectos e suas interacções para desenhar e desenvolver aplicações., 42

PDA PDA significa Assistente Pessoal Digital, é um computador de dimensões reduzidas., 42

Pocket Internet Explorer Browser disponibilizado pelo sistema operativo Windows Mobile, utilizado em dispositivos móveis para aceder à Internet., 42

PocketPC Especificação da Microsoft de um dispositivo móvel (PDA)., 42

POJO Simples objecto Java. Esta designação é utilizada para classificar os objectos java que mapeiam a estrutura relacional., 42

Serviço Classe que implementa as regras do Negócio (Camada de Negócio)., 42

Servlet Componente Java que permite adicionar conteúdo dinâmico a um servidor de aplicações., 42

SGBD Aplicação responsável por gerir base de dados. O SGBD disponibiliza a interface para se realizar operações sobre a base de dados., 42

SIAG Solução informática de apoio à gestão desenvolvida pela GEDI., 42

Spring Framework utilizada para gerir os objectos responsáveis por implementar a camada de negócio., 42

Struts Framework utilizada no desenvolvimento de aplicações web. Orienta o desenvolvimento no sentido de uma arquitectura Modelo-Vista-Controlo., 42

XML É uma linguagem de marcação capaz de descrever diversos tipos de dados., 42

Índice

Bibliografia

[1] http://tiles.apache.org/.

[2] http://en.wikipedia.org/wiki/.

[3] Agile manifesto. http://agilemanifesto.org/.

[4] Designing web sites for the internet explorer for pocket pc. http://msdn2.microsoft.com/en-us/library/ms838316.aspx.

[5] Hibernate tutorials. http://hibernate.javabeat.net/tutorials/.

[6] Javaserver pages technology - frequently asked questions. http://java.sun.com/products/jsp/faq.html.

[7] Model-view-controller. http://java.sun.com/blueprints/patterns/MVC-detailed.html.

[8] Model-view-controller pattern. http://www.enode.com/x/markup/tutorial/mvc.html.

[9] Top reasons to use mysql. http://www.mysql.com/why-mysql/topreasons.html.

[10] Christian Bauer and Gavin King. *Hibernate in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.

[11] Peggy Chen. Oracle application server wireless: Mobilizing your enterprise. *edoc*, 00:0016, 2003.

[12] Philip Darringer. Oracle application server wireless: Extending the reach of enterprise applications. *edoc*, 00:0017, 2005.

[13] Mauricio Freitas. Pocket pc browser comparison. http://www.geekzone.co.nz/content.asp?contentid=505.

[14] Ted Husted, Cedric Dumoulin, George Franciscus, and David Winterfeldt. *Struts in Action: Building web applications with the leading Java framework)*. Manning Publications Co., Greenwich, CT, USA, 2002.

[15] Carnegie Mellon Software Engineering Institute. STR technology descriptions. `http://www.sei.cmu.edu/str/descriptions/`.

[16] Rod Johnson, Juergen Hoeller, and Alef Arendsen et al. *The Spring Framework - Reference Documentation.* `http://www.springframework.org/docs/reference/index.html`.

[17] BEA Systems. Bea weblogic mobility server 3.6 - features and benefits. `http://www.bea.com/framework.jsp?CNT=features.htm&FP=/content/products/weblogic/mobility/`.